



GREENPLUM SUMMIT by Pivotal.

AT POSTGRESCONF | NEW YORK | #SCALEMATTERS

Pivotal.

Agile Data Science on Greenplum Using Airflow

Ambarish Joshi - Data Scientist

Aditya Padhye - Data Engineer

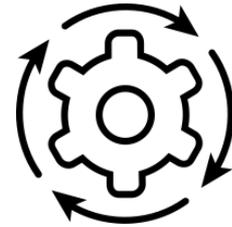


“**Agile software development** refers to a group of **software development** methodologies based on **iterative development**, where requirements and solutions evolve through collaboration between self-organizing cross-functional teams.”

Data Science Phases



Discovery Phase



Operationalization (O16n) Phase

Data Science Phases



Discovery Phase

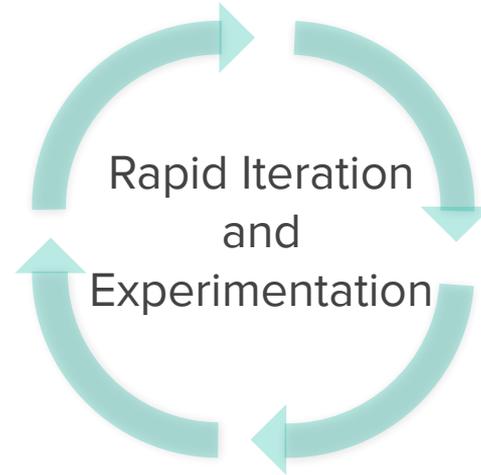
- ✓ Data exploration & cleaning
- ✓ Feature engineering
- ✓ Model Building
- ✓ Model Evaluation

Data Science Phases - Agility



Discovery Phase

- ✓ Data exploration & cleaning
- ✓ Feature engineering
- ✓ Model Building
- ✓ Model Evaluation

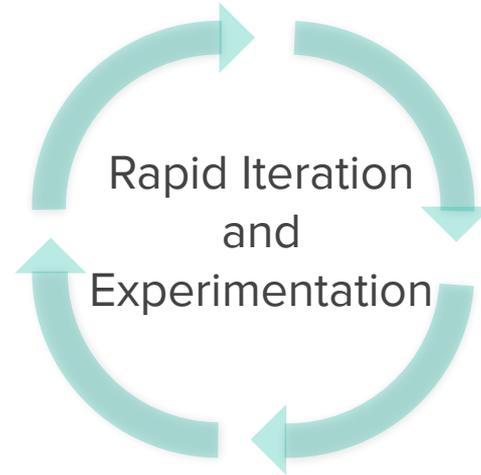


Data Science Phases - Agility



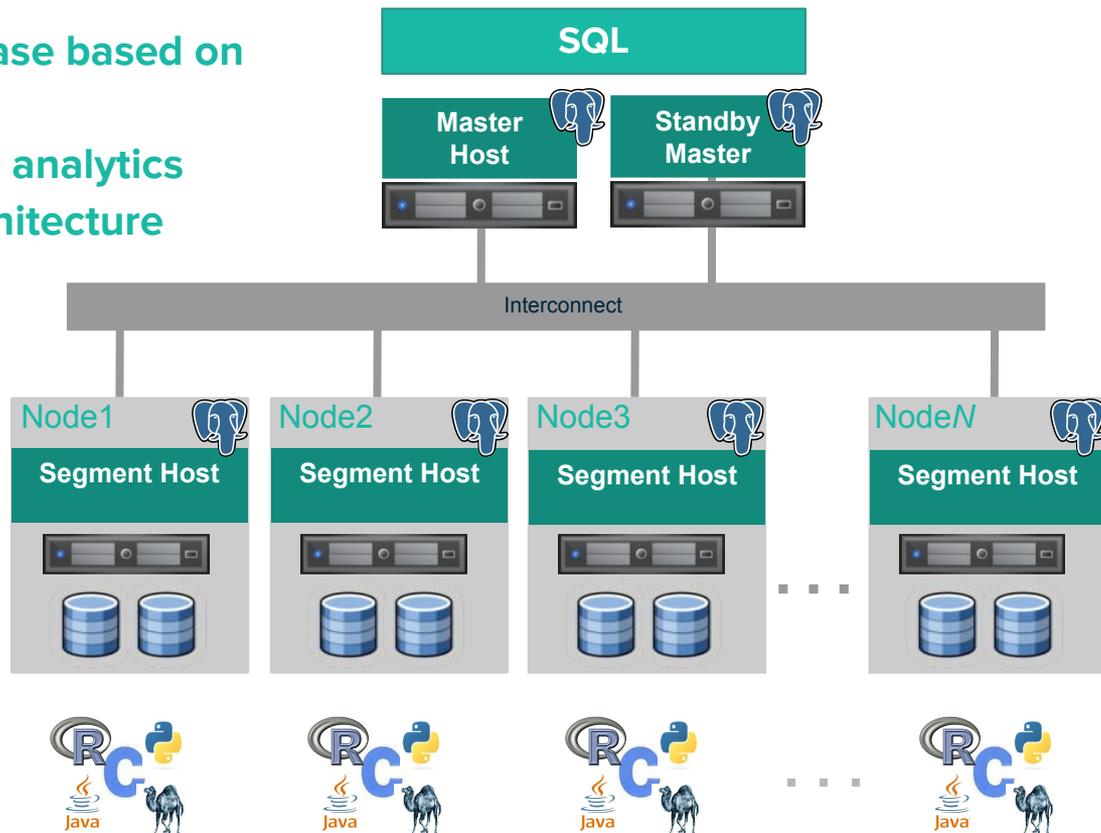
Discovery Phase

- ✓ Data exploration & cleaning
- ✓ Feature engineering
- ✓ Model Building
- ✓ Model Evaluation



Greenplum Database

- MPP database based on Postgres
- In database analytics
- Parallel architecture



Jupyter Notebooks

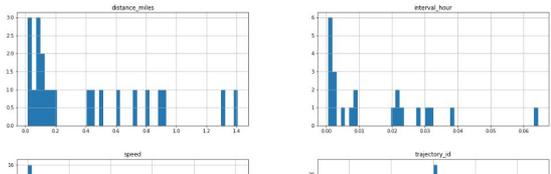
Jupyter Geolife Airflow Last Checkpoint: 20 hours ago (autosaved) Python 3.0

File Edit View Insert Cell Kernel Widgets Help Not Trusted Python 3.0

```
75% 501.0 0.691260 0.023056 53.589242
max 501.0 1.406879 0.064444 2532.381804
```

In [15]: `rcParams['figure.figsize'] = 20, 10
df.hist(bins=50)`

Out[15]: array([[<matplotlib.axes._subplots.AxesSubplot object at 0x117e3ee80>,
<matplotlib.axes._subplots.AxesSubplot object at 0x11811ee49>],
[<matplotlib.axes._subplots.AxesSubplot object at 0x11813f470>,
<matplotlib.axes._subplots.AxesSubplot object at 0x118162b00>]],
dtype=object)



In [16]: `df[df.speed > 1000]`

Out[16]:

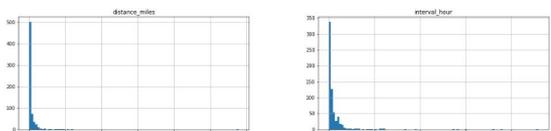
uid	trajectory_id	mode	pt
4	117	501 train	0101000020E61000005BFAF99034155D40545227A08900... 0101000020E61000
16	117	501 train	0101000020E6100000C649312DD7125D400E3C9827EA07... 0101000020E61000

In [23]: `df_train = %sql select * from exp.trajectory_label_speed where mode = 'train'
* postgresql://airflow_user:***@172.16.223.128/airflow_test
662 rows affected.`

In [24]: `df_train = df_train.DataFrame()`

In [27]: `df_train.hist(bins=100)`

Out[27]: array([[<matplotlib.axes._subplots.AxesSubplot object at 0x118c81cf9>,
<matplotlib.axes._subplots.AxesSubplot object at 0x119175748>],
[<matplotlib.axes._subplots.AxesSubplot object at 0x119194d68>,
<matplotlib.axes._subplots.AxesSubplot object at 0x1191bf400>]],
dtype=object)



Jupyter Modeling (unsaved changes) Python 3.0

File Edit View Insert Cell Kernel Widgets Help Trusted Python 3.0

```
In [1]: import matplotlib.pyplot as plt
import pandas as pd
import numpy as np
import matplotlib.pyplot as plt
import seaborn as sns
from tsfresh import extract_features, extract_relevant_features, select_features
from tsfresh.utilities.dataframe_functions import impute
from sklearn.cross_validation import train_test_split
from sklearn.metrics import classification_report, confusion_matrix

/Users/ajoshi/anaconda3/envs/gpdb-airflow/lib/python3.6/site-packages/sklearn/cross_vali
dation.py:41: DeprecationWarning: This module was deprecated in version 0.18 in favor of
the model_selection module into which all the refactored classes and functions are move
d. Also note that the interface of the new CV iterators are different from that of this
module. This module will be removed in 0.20.
"this module will be removed in 0.20.", DeprecationWarning)
```

In [2]: `import os
GPDB_HOST = os.environ['GPDB_HOST']`

In [3]: `%load_ext sql
%sql postgresql://airflow_user:airflow@GPDB_HOST/airflow_test`

Out[3]: 'Connected: airflow_user@airflow_test'

Create tsfresh features

```
In [4]: %%sql
drop function if exists tsfresh_features(
text[],
timestamp[],
float[],
float[],
float[]
);

create or replace function tsfresh_features(
trajectory_id text[],
ttime timestamp[],
distance_miles float[],
interval_hour float[],
speed float[]
)
returns setof ts_features
as
$$
import pandas as pd
import numpy as np
from tsfresh import extract_features
from tsfresh.utilities.dataframe_functions import impute
from tsfresh.feature_extraction import ComprehensiveFCParameters, MinimalFCParameters

df = pd.DataFrame({'id': trajectory_id,
'time': ttime,
'distance_miles': distance_miles,
'interval_hour': interval_hour,
'speed': speed})

extraction_settings = MinimalFCParameters()

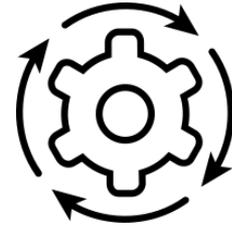
X = extract_features(df, column_id='id', column_sort='time',
default_fc_parameters=extraction_settings,
impute_function=impute)

X = X.reset_index()
X = X.melt(id_vars=['id'])
```

Data Science Phases

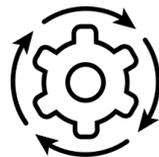


Discovery Phase



Operationalization (O16n) Phase

Data Science Phases



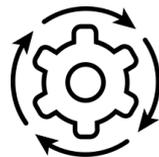
Operationalization (O16n) Phase

- ✓ Data Pipelines
- ✓ Testing
- ✓ Monitoring
- APIs to consume model output

Data Science Phases - Agility

- ✓ Automated manageable pipelines
- ✓ Testing with CI
- ✓ Monitoring to react to Failures

Madlib Flow Talk by Frank and Sridhar



Operationalization (O16n) Phase

- ✓ Data Pipelines
- ✓ Testing
- ✓ Monitoring

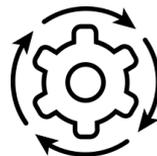
APIs to consume model output

Data Science Phases - Agility



- ✓ Automated manageable pipelines
- ✓ Testing with CI
- ✓ Monitoring to react to Failures

Madlib Flow Talk by Frank and Sridhar



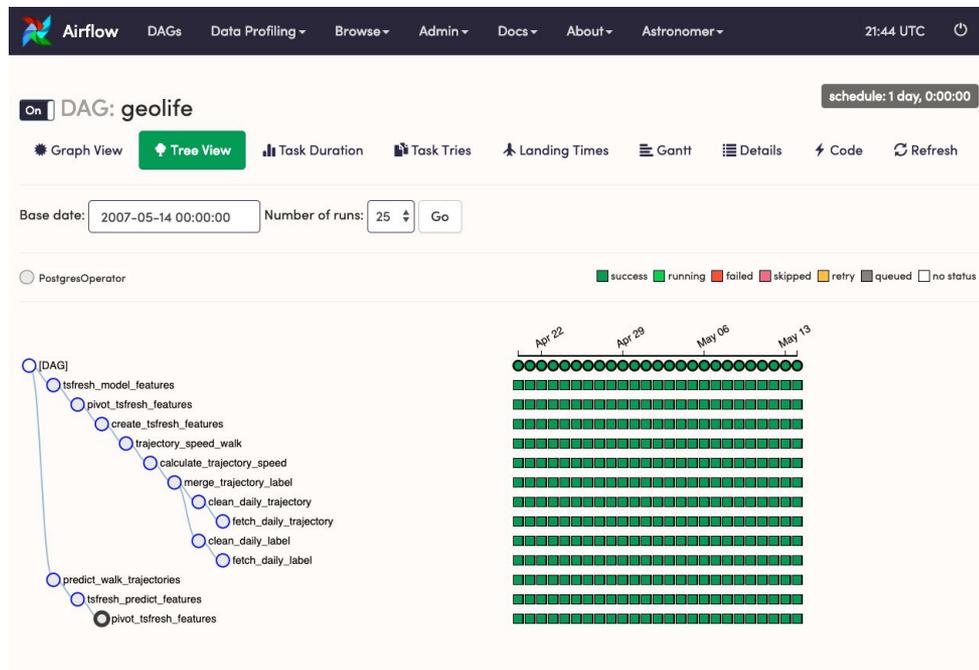
Operationalization (O16n) Phase

- ✓ Data Pipelines
- ✓ Testing
- ✓ Monitoring

APIs to consume model output

Airflow

- Apache Project spun out of Airbnb
- “Airflow is a platform to programmatically author, schedule and monitor workflows.”



Data Science Use-Case

- **The Data**
 - **Time-series trajectories** with **latitude** and **longitude** of location.
 - Subset of trajectories are labeled as **walk / not walk**
- **Our Model**
 - Build **Classification model** using labelled data to identify if new unlabeled trajectories are walk or not walk

Example trajectories



Example data

We have mode labels of walk and not walk only for subset of incoming daily trajectories

Example trajectory data

uid	latitude	longitude	tdate	ttime
020	39.97445333333333	116.3021633333333	2011-08-25	14:38:25
020	39.97445	116.302165	2011-08-25	14:38:26
020	39.97460166666667	116.3020733333333	2011-08-25	14:39:22
020	39.97473	116.3020666666667	2011-08-25	14:39:23
020	39.97481166666667	116.30207	2011-08-25	14:39:24

Example label data

uid	start_date	start_time	end_date	end_time	mode
020	2011-08-27	06:13:01	2011-08-27	08:01:37	walk
020	2011-08-27	09:34:43	2011-08-27	14:50:30	walk
020	2011-08-27	14:50:31	2011-08-27	15:01:58	bus
020	2011-08-27	15:01:59	2011-08-27	15:31:43	walk
020	2011-08-28	04:33:31	2011-08-28	04:44:25	walk

Discovery phase → Operationalization phase

After every model iteration we check if the model is viable

- Check the quantitative metrics of the model like AUC, ROC curve, accuracy etc
- Check the qualitative results of the model and if it make sense to a subject matter expert

Once we are convinced that the model is both quantitatively and qualitatively viable we can move to the Operationalization phase

Discovery phase → Operationalization phase

Example of code from the discovery phase which is converted into a task script

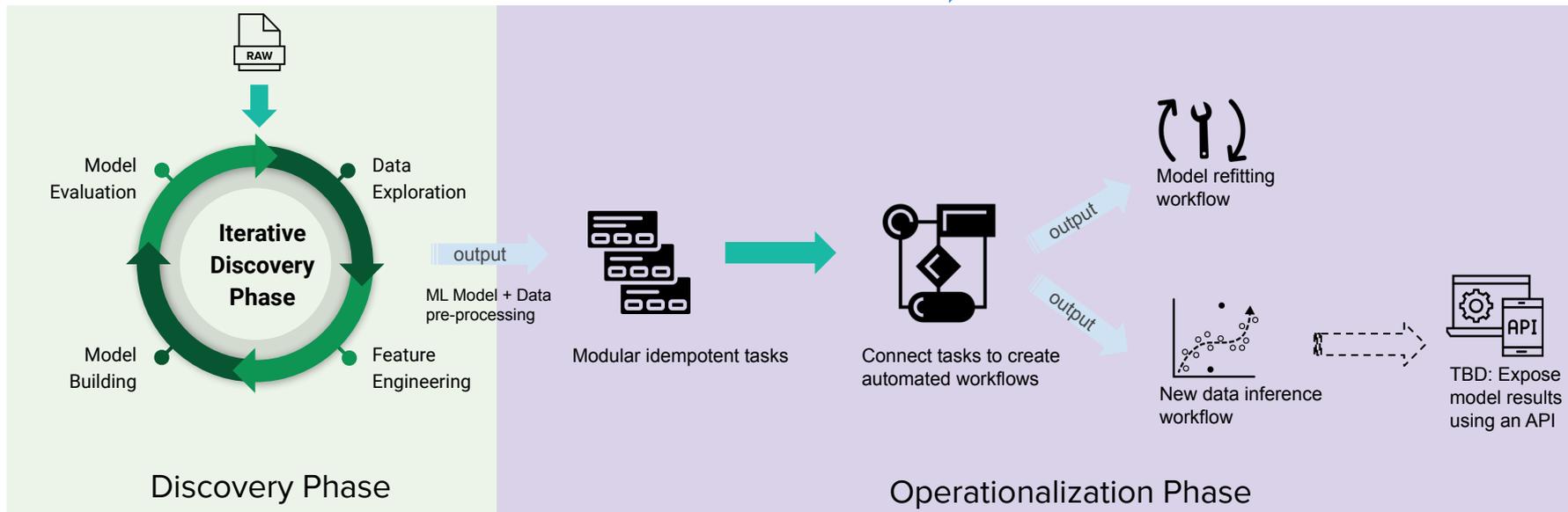
```
Jupyter Geolife Airflow (autosaved) Logout
File Edit View Insert Cell Kernel Widgets Help Not Trusted Python 3 O
Code
calculate speed
In [53]: %%sql
drop table if exists exp.trajectory_label_speed;
create table exp.trajectory_label_speed
as
with lead_trajectory as (
select *,
    lead(latitude) over(partition by trajectory_id order by ttimestamp) as lead_lat,
    lead(longitude) over(partition by trajectory_id order by ttimestamp) as lead_long,
    lead(ttimestamp) over(partition by trajectory_id order by ttimestamp) as lead_ttimestamp
from exp.trajectory_label
),
t2 as (
select uid,
    trajectory_id,
    mode,
    pt,
    ST_SetSRID(st_point(lead_long, lead_lat),4326) as lead_pt,
    tdate,
    ttime,
    ttimestamp,
    lead_ttimestamp
from lead_trajectory
),
t3 as (
select *,
    st_distance(st_transform(pt, 2163) , st_transform(lead_pt, 2163)) / 1609.34 as dis
    EXTRACT(EPOCH FROM (lead_ttimestamp - ttimestamp)) / 3600.0 as interval_hour
from t2
where lead_ttimestamp != ttimestamp --removing divide by zero error
)
select *,
    distance_miles / interval_hour as speed
from t3
distributed by (uid, trajectory_id)

* postgresql://airflow_user:***@172.16.223.128/airflow_test
Done.
35177 rows affected.
```



```
calculate_trajectory_speed.sql
1 alter table geolife.geolife_trajectory_label_speed drop partition if exists p{{ ds_nodash }};
2 alter table geolife.geolife_trajectory_label_speed add partition p{{ ds_nodash }}
3 values (date '{{ ds }}');
4
5 -- Calculate distance, interval and speed between 2 consecutive gps location points
6 insert into geolife.geolife_trajectory_label_speed
7 with lead_trajectory as (
8 select *,
9     lead(latitude) over(partition by trajectory_id order by ttimestamp) as lead_lat,
10    lead(longitude) over(partition by trajectory_id order by ttimestamp) as lead_long,
11    lead(ttimestamp) over(partition by trajectory_id order by ttimestamp) as lead_ttimestamp
12 from geolife.geolife_trajectory_label_clean
13 where tdate = '{{ ds }}'
14 ),
15 t2 as (
16 select uid,
17    trajectory_id,
18    mode,
19    pt,
20    ST_SetSRID(st_point(lead_long, lead_lat),4326) as lead_pt,
21    tdate,
22    ttime,
23    ttimestamp,
24    lead_ttimestamp
25 from lead_trajectory
26 ),
27 t3 as (
28 select *,
29     st_distance(st_transform(pt, 2163) , st_transform(lead_pt, 2163)) / 1609.34 as distance_miles,
30     EXTRACT(EPOCH FROM (lead_ttimestamp - ttimestamp)) / 3600.0 as interval_hour
31 from t2
32 where lead_ttimestamp != ttimestamp --removing divide by zero error
33 )
34 select *,
35     distance_miles / interval_hour as speed
36 from t3;
37
```

Architecture overview

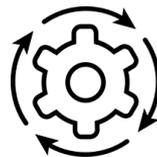


Data Science Phases - Agility



- ✓ Automated manageable Pipelines
- ✓ Testing with CI/CD
- ✓ Monitoring to React to Failures

Madlib Flow Talk by Frank and Sridhar

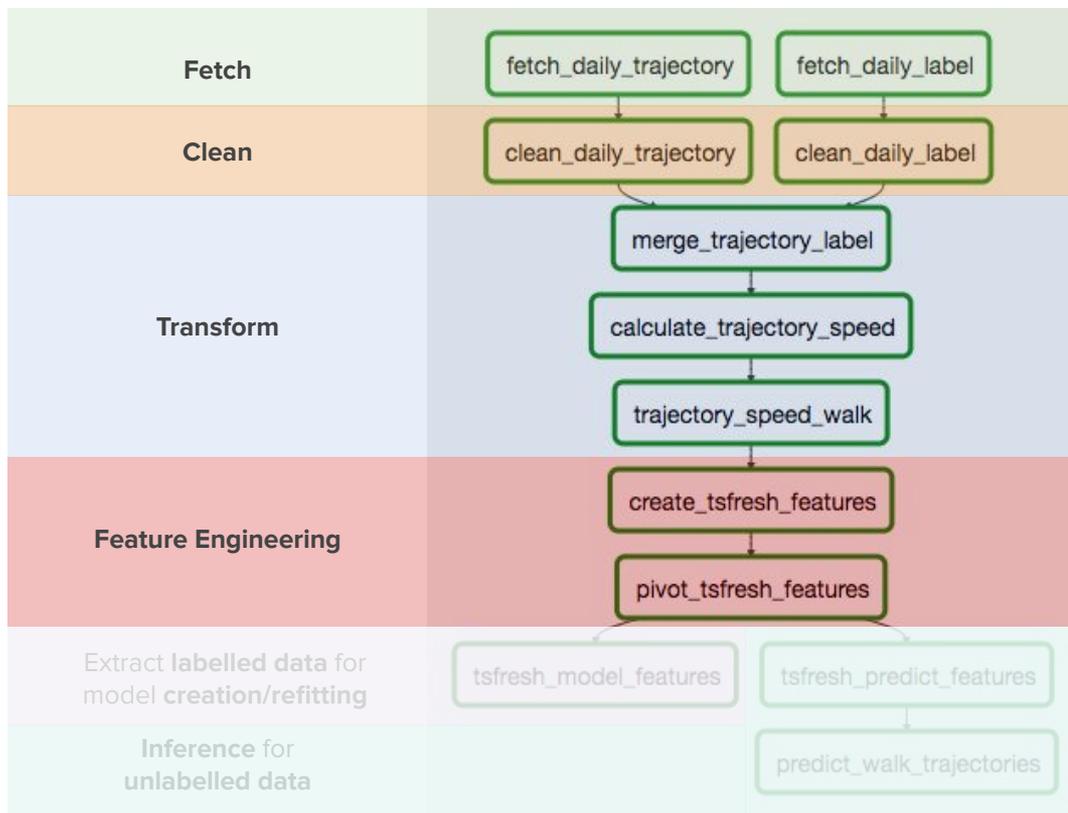


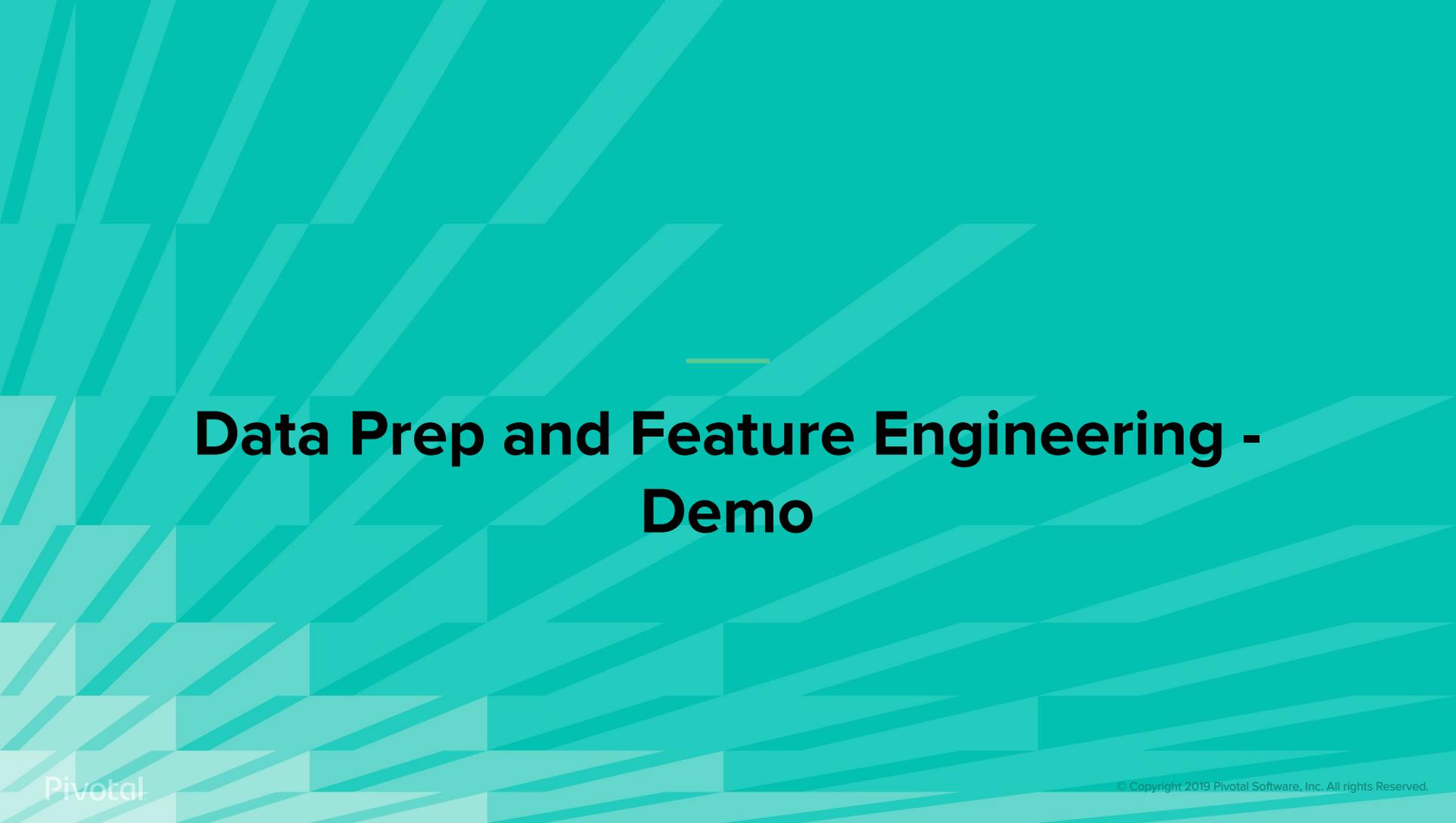
Operationalization (O16n) Phase

- ✓ Pipelines
- ✓ Testing
- ✓ Monitoring

APIs to consume model output

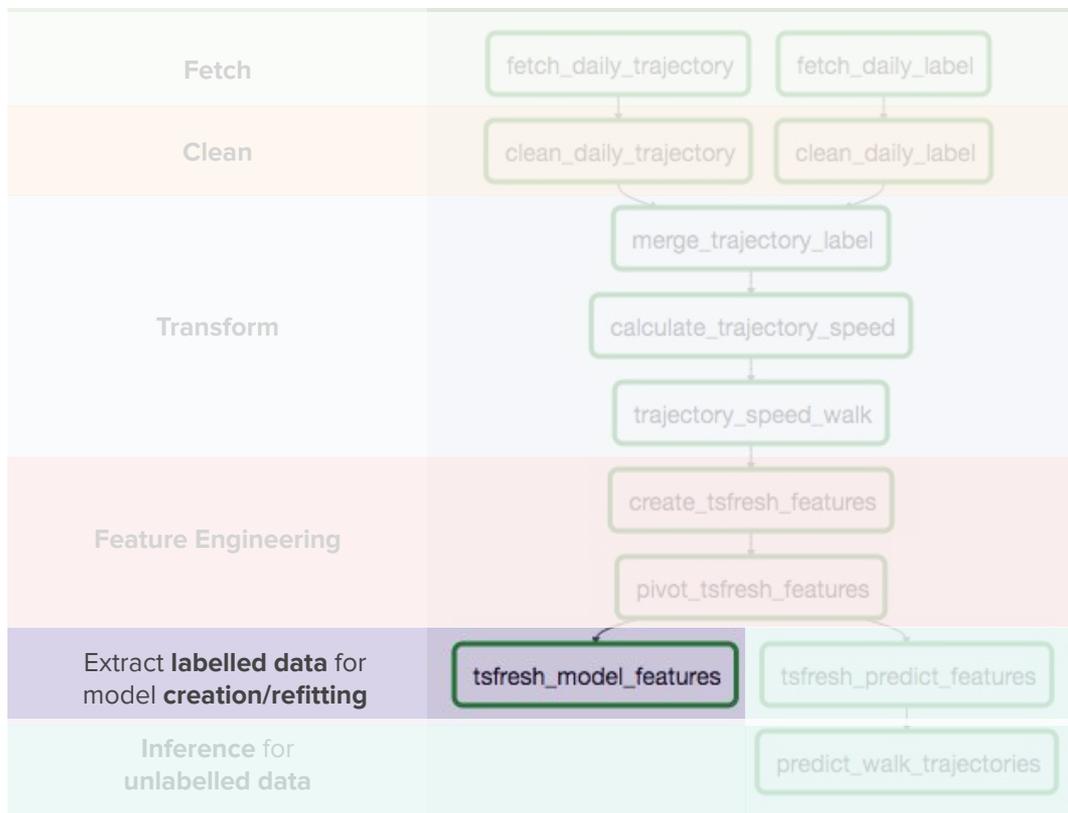
Data Prep and Feature Engineering





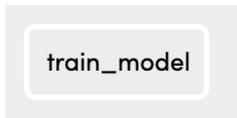
Data Prep and Feature Engineering - Demo

Model Training



Model Training

- This DAG has a single task for model training
- In this task we split the data into train and test samples, train the model, evaluate the model and capture the accuracy, auc and model tables.
- We want all of the above to run at the same time

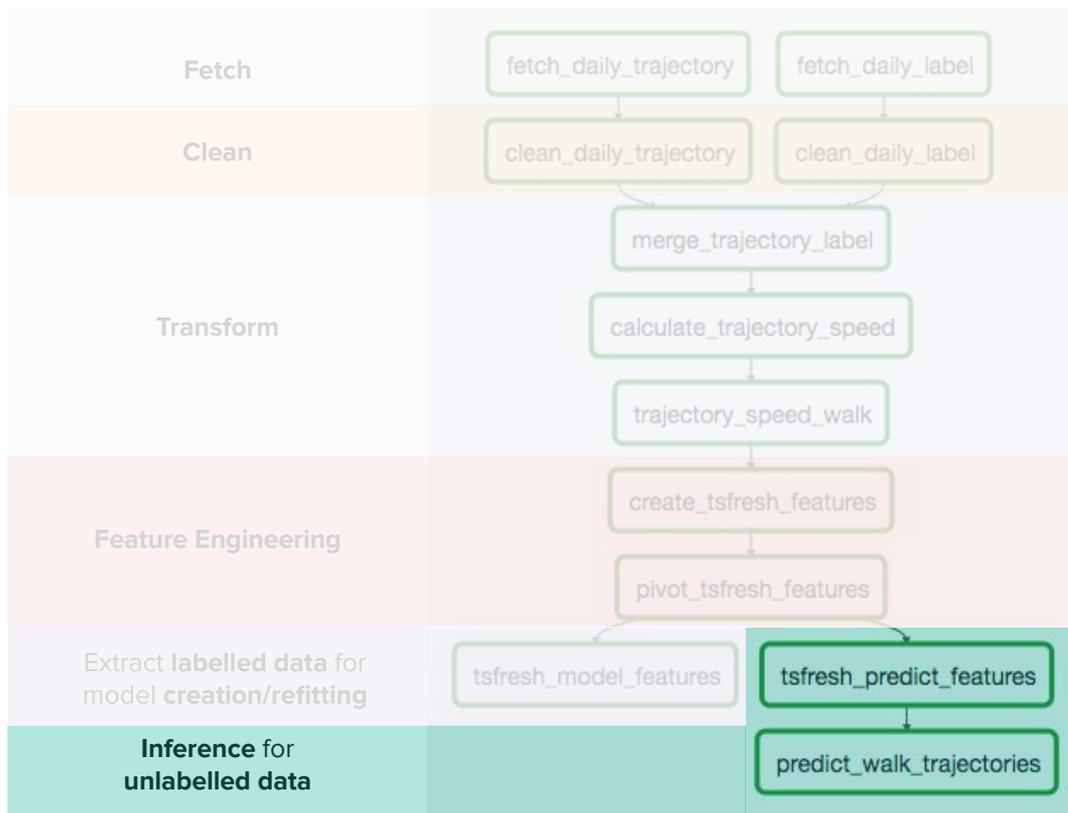


```
SELECT madlib.train_test_split(  
    'geolife.tsfresh_model_features', -- Source table  
    'geolife.features_walk_{{ds_nodash}}', -- Output table  
    0.8, -- Sample proportion  
    0.2, -- Sample proportion  
    NULL, -- Strata definition  
    NULL, -- Columns to output  
    FALSE, -- Sample without replacement  
    TRUE); -- Separate output tables  
  
-- build a random forest model using madlib  
  
DROP TABLE IF EXISTS geolife.rf_walk_{{ds_nodash}}_output, geolife.rf_walk_{{ds_nodash}}_output;  
SELECT madlib.forest_train('geolife.features_walk_{{ds_nodash}}_train', -- source table  
    'geolife.rf_walk_{{ds_nodash}}_output', -- output model table  
    'id', -- id column  
    'label', -- response  
    '*', -- features  
    'tdate', -- exclude columns  
    NULL, -- grouping columns  
    20::integer, -- number of trees  
    2::integer, -- number of random features  
    TRUE::boolean, -- variable importance  
    1::integer, -- num_permutations  
    8::integer, -- max depth  
    3::integer, -- min split  
    1::integer, -- min bucket  
    10::integer -- number of splits per continuous variable  
);  
  
-- Evaluate the built model  
  
DROP TABLE IF EXISTS geolife.rf_walk_{{ds_nodash}}_results;  
SELECT madlib.forest_predict('geolife.rf_walk_{{ds_nodash}}_output', -- tree model  
    'geolife.features_walk_{{ds_nodash}}_test', -- new  
    'geolife.rf_walk_{{ds_nodash}}_results') ;--, -- output table  
    --'prob');  
  
-- Capture model results  
  
drop table if exists geolife.walk_{{ds_nodash}}_result;  
create table geolife.walk_{{ds_nodash}}_result  
as  
with t as (  
select id,  
    case when label = True then 1.0 else 0.0 end as obs  
from geolife.features_walk_{{ds_nodash}}_test
```



Model Training - Demo

Model Scoring



Model Scoring

- The unlabeled data which is extracted from the features table is scored in this DAG
- We first check if any model has been built
- If there is a model so we score the data (inference)

```
DO
$do$
DECLARE
tablename character varying(255);
BEGIN
IF (select count(*) from
(select 1 from geolife.tsfresh_predict_features{{ds_nodash}} limit 1) as t) > 0
and
(select count(*) from (select 1 from geolife.models_metadata limit 1) as p) > 0
THEN
tablename := (select model_tabname
from geolife.models_metadata
order by mdate
limit 1);

DROP TABLE IF EXISTS prediction_results;
PERFORM madlib.forest_predict(tabname,
'geolife.tsfresh_predict_features{{ds_nodash}}',
'geolife.walk_prediction_results{{ds_nodash}}',
'response');

insert into geolife.walk_prediction_results
select *,
regexp_replace(id, '^\.[0-9-]{10}_.*$', E'\1'):date as tdate
from geolife.walk_prediction_results{{ds_nodash}};

END IF;
END
$do$

drop table if exists geolife.tsfresh_predict_features{{ds_nodash}};
drop table if exists geolife.walk_prediction_results{{ds_nodash}};
```



Model Scoring - Demo

Model Re-Training

- Daily we get some more labeled data, once we have accumulated enough labeled data we can retrain the model for better accuracy
- We have scheduled model re-training monthly



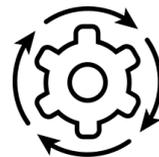
Model Re-Training - Demo

Data Science Phases - Agility



- ✓ Automated manageable Pipelines
- ✓ Testing with CI/CD
- ✓ Monitoring to React to Failures

Madlib Flow Talk by Frank and Jarrod



Operationalization (O16n) Phase

- ✓ Pipelines
- ✓ Testing
- ✓ Monitoring

APIs to consume model output

Testing with CI/CD

- **Testing Data Pipelines is hard**
- **Test Coverage (Test Tasks vs Test DAGs)**
- **Testing as part of the CI/CD**



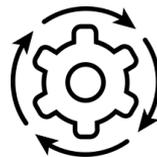
Testing with CI/CD - Demo

Data Science Phases - Agility



- ✓ Automated manageable Pipelines
- ✓ Testing with CI/CD
- ✓ Monitoring to React to Failures

Madlib Flow Talk by Frank and Sridhar



Operationalization (O16n) Phase

- ✓ Pipelines
- ✓ Testing
- ✓ Monitoring

APIs to consume model output

Monitoring and Error Fixing

- Monitoring and error fixing is big part of responsive data pipelines
- Ability to quickly identify what is failing, why it is failing and fixing it with minimum lead time is crucial
- In this demo we will showcase an error fixing case

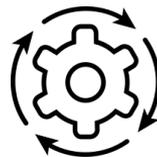
Monitoring and Error Fixing - Demo

Data Science Phases - Agility



- ✓ Automated manageable Pipelines
- ✓ Testing with CI/CD
- ✓ Monitoring to React to Failures

Madlib Flow Talk by Frank and Sridhar



Operationalization (O16n) Phase

- ✓ Pipelines
- ✓ Testing
- ✓ Monitoring

APIs to consume model output

Conclusion

- ✓ **Greenplum and Jupyter notebooks provides a set of tools to do Agile Data Science during discovery phase**
- ✓ **Greenplum along with Airflow and Circle CI is very effective to do Agile Data Science during the operationalization phase**

A group of people in a meeting room. A man on the left is pointing at a whiteboard. Several people are sitting on stools, listening. A man on the right is standing with his arms crossed, looking towards the group. The room has a whiteboard, a desk, and a window in the background.

Questions



**“We partner to help you
compete, grow, and transform.”**